5. RFID Tutorial: Define visit type and learning

Gabrielle Davidson - School of Biological Sciences, University of East Anglia

2024-08-22

5. Get set up

Open R Studio

Check R packages installed

```
#load the following packages
library(tidyverse) #some of the packages below are included in tidyverse
library(rmarkdown)
library(dplyr)
library(tidyr)
library(lubridate)
library(ggplot2)
library(stringi)
library(magrittr) #may not need this

# These are new and need to be installed
library(writexl)
library(readxl)
library(zoo) #rolling function
```

Download resources necessary for the workshop

Download the resources from **my github repository (https://github.com/DrGLDavidson/RFID-workshop)**

5.1 import your dataset to your environment from your working directory.

```
#clear the global environment so we don't have any conflicts with the next steps

rm(list = ls(all.names = TRUE))

#choose the appropriate working directory

setwd("F:/RWorkspace/GitHub/RFID-workshop/data/outputFiles")

#call your most recent dataset

df<-read.delim("Masterdf_noRepeats.txt", header=TRUE)
head(df)
```

```
##         date            dateTime Hmsec     ID Event Channel    Dur Clks   Freq
## 1 2022-02-04 2022-02-04 10:01:13   467 c1935    146       0      0   32 126464
## 2 2022-02-04 2022-02-04 11:06:19   516 c1931  15597       0 127943   26      0
## 3 2022-02-04 2022-02-04 11:06:32   770 c1935    147       0      6   39 126208
## 4 2022-02-04 2022-02-04 11:17:36   871 c1935    148       0 105573   51 126464
## 5 2022-02-04 2022-02-04 11:17:40   123 c1935    148       0 121015   27 126208
## 6 2022-02-04 2022-02-04 11:17:48   543 c1931  15598       0      0   39      0
##   Edges Reps Type  TagID_hex feeder dateRinged timeRinged btoRingType   btoID
## 1   264    2    1 01103FC949    F01 09/01/2021      11:03           R AKL0680
## 2   442    1    1 01103F7DB1    F02 03/10/2021      10:30           R AAJ5894
## 3   426    3    1 01103F7DB1    F01 03/10/2021      10:30           R AAJ5894
## 4  1168    5    1 01103F3BED    F01 17/10/2021      10:09           R AAJ5895
## 5   174    1    1 01103F3BED    F01 17/10/2021      10:09           R AAJ5895
## 6   388    3    1 01103FE3B3    F02 07/11/2021      12:15           R AJT8118
##   species pitTYPE age sex wing weight timeSincePreviousVisit
## 1      BT       R   5   M   64   11.7             firstVisit
## 2      GT       R   3   F   73   16.3             firstVisit
## 3      GT       R   3   F   73   16.3             firstVisit
## 4      GT       R   3   M   75   18.5             firstVisit
## 5      GT       R   3   M   75   18.5                      4
## 6      GT       R   3   F   72   18.1             firstVisit
```

```
names(df)
```

```
##  [1] "date"                  "dateTime"              "Hmsec"
##  [4] "ID"                    "Event"                 "Channel"
##  [7] "Dur"                   "Clks"                  "Freq"
## [10] "Edges"                 "Reps"                  "Type"
## [13] "TagID_hex"             "feeder"                "dateRinged"
## [16] "timeRinged"            "btoRingType"           "btoID"
## [19] "species"               "pitTYPE"               "age"
## [22] "sex"                   "wing"                  "weight"
## [25] "timeSincePreviousVisit"
```

5.2 Quantifying correct and incorrect visits (i.e. if a bird landed on its assigned feeder according to ListA logic)

```
#how many individual tags (birds) in our dataset
length(unique(df$TagID_hex))
```

In practice, you will have (or create) a file for each bird indicating which feeder opens for which bird. This is often referred to as “ListA” because this is what the logic is called in the logger.ini file associated with the computerised feeder.

For the purpose of this tutorial, we create a random vector of data to indicate which feeder opens for a given bird. Draw 44 samples from the integers 1:3 with replacement (i.e. each number can be redrawn), and attach that to unique bird IDs.

```
assignedFeeder<-sample(x = 1:3, size = 44, replace = TRUE)
#unique birds in df
TagID_hex<-unique(df$TagID_hex)

df2<-cbind(TagID_hex, assignedFeeder)
df2<-as.data.frame(df2)

#change values in assignedFeeder to match feeder values in df.

df3 <- df2 %>%
  mutate(assignedFeeder = recode(assignedFeeder,
                            "1" = "F01",
                            "2" = "F02",
                            "3" = "F03"))

df4 <- merge(df, df3, by = "TagID_hex", all = TRUE)

#create a vector of whether values in df4$feeder are identical to df4$assignedFeeder and return
TRUE or FALSE
correctVisit <- c()
for (i in 1:nrow(df4)) {
  correctVisit[i] <- identical(df4$feeder[i], df4$assignedFeeder[i])
}

#add the vector to df4
df4$correctVisit <- correctVisit

#create a binary variable column for downstream statistical analyses and figures

df4$correctVisit<-as.character(df4$correctVisit)

df5 <- df4 %>%
  mutate(correctVisitBinary = recode(correctVisit,
                            "TRUE" = "1",
                            "FALSE" = "0"))


write.table(df5, file = "Masterdf_correctVisits.txt",sep="\t",row.names=FALSE)
```

5.3 Quantifying learning (i.e. if a bird visits its assigned feeder at a given criterion).

In this example, we will use the criterion of 80% correct visits, across a minimum of 20 visits.

First, define participants as birds that have visited at least 20 times.

```
#filter out birds that have visited at least 20 times

class(df5$correctVisitBinary)
df5$correctVisitBinary<-as.numeric(df5$correctVisitBinary)


df6 <- df5 %>%
  group_by(TagID_hex) %>%
  mutate(visit_count = n()) %>%  #what variable is visit count
  filter(visit_count >= 20) %>%
  ungroup() %>%
  select(-visit_count)
```

Below you will use a function for quantifying how many visits until a bird has met “criterion”. This function was created by Vildan Acar.

We define criterion as 80% correct over 20 visits.

Note that the code also includes a filtering step of birds having to have had at least 20 visits, making the above code redundant. There is no harm in running both. And the above code may be useful if you’re interested in participation only (rather than learning)

Description of the function **success_chec()** written by Vildan Acar.

“correct” is a vector that contains binary values (1 for correct visits, 0 for false visits). Each element should correspond to a visit. The success_check function determines the first instance when a bird meets a specific learning criterion based on visits. For the function: At least 20 consecutive visits are required. The first visit within these 20 consecutive visits must be a correct visit (indicated by window[1] == 1) and 80% or more of these 20 visits must be correct (sum(window)/ 20 >= 0.8) for the criterion to be met. 19 refers to a count related to 20 consecutive visits.

```
success_check <- function(correct ){
  n <- length(correct)  #what variable is correct
 if (n < 20){
   return(NA)
 }
  for (i in 1:(n-19)) {
    window <- correct[i: (i+ 19)]
    if(window[1] == 1 && sum(window)/ 20 >= 0.8){
      return(i+19)
    }

  }
  return(NA)
}

#### birds visiting >20 ####

#apply criterion check function
results20 <- df5 %>%
  group_by(TagID_hex) %>%
  summarise(criterion = success_check(correctVisitBinary)) #here "correctVisitBinary" is the co
lumn with 0 or 1 values and should be numerical
head(results20)
view(results20)
```

You will see there are many NAs. This is because most birds have yet to learn under our criteria.

To demonstrate, you could alter the values in the function so it is less stringent (and simply because our dataset is small, birds have yet to learn). The participation will be 5 visits, and 80% correct in 5 visits

```
success_check2 <- function(correct ){
  n <- length(correct)  #what variable is correct
 if (n < 5){
   return(NA)
 }
  for (i in 1:(n-4)) {
    window <- correct[i: (i+ 4)]
    if(window[1] == 1 && sum(window)/ 5 >= 0.8){
      return(i+4)
    }

  }
  return(NA)
}

#apply criterion check function
results5 <- df5 %>%
  group_by(TagID_hex) %>%
  summarise(criterion = success_check2(correctVisitBinary)) #notice the function is changed her
e to "success_check2"
head(results5)
view(results5)
```

While there are still a lot of NAs, but you can see there are more birds that learned, and the number of visits it took to learn. As your dataset gets larger, so will number of birds meeting criterion.

These file/tally will be useful data to keep track of what is happening with the experiment every time you generate data.

If you wish to save this data, you could do that now.

```
path_out = 'F:/RWorkspace/GitHub/RFID-workshop/data/outputFiles'
write.csv(results5, file.path(path_out,'birdsReachingCriterion_5visits.csv'))
```

END OF 5. RFID Tutorial: Define visit type and learning